Dense-Resolution Network for Point Cloud Classification and Segmentation

Shi Qiu[1,2], Saeed Anwar[1,2] and Nick Barnes[1]
[1]Australian National University, [2]Data61-CSIRO, Australia
{shi.qiu, saeed.anwar}@data61.csiro.au, nick.barnes@anu.edu.au

Abstract

Point cloud analysis is attracting attention from Artificial Intelligence research since it can be widely used in applications such as robotics, Augmented Reality, self-driving. However, it is always challenging due to irregularities, unorderedness, and sparsity. In this article, we propose a novel network named Dense-Resolution Network (DRNet) for point cloud analysis. Our DRNet is designed to learn local point features from the point cloud in different resolutions. In order to learn local point groups more effectively, we present a novel grouping method for local neighborhood searching and an error-minimizing module for capturing local features. In addition to validating the network on widely used point cloud segmentation and classification benchmarks, we also test and visualize the performance of the components. Comparing with other state-of-the-art methods, our network shows superiority on ModelNet40, ShapeNet synthetic and ScanObjectNN real point cloud datasets.

1. Introduction

With the help of rapid progress in 3D sensing technology, an increasing number of researchers are now focusing on 3D point clouds. Different from complex 3D data *e.g.*, mesh and volumetric data, point clouds have a simpler data format. Typically, point clouds are easier to collect using different types of scanners [3] with specific algorithms: *e.g.*, LiDAR scanners [12] and Simultaneous localization and mapping (SLAM) algorithms. Traditional algorithms addressing point cloud learning [32, 24, 31, 37] used to estimate geometric information and capture indirect clues utilizing complicated models. In contrast, deep learning models provide explicit and effective data-driven approaches to acquire information from 3D point cloud data, leveraging Convolutional Neural Networks (CNN).

In general, CNN-related methods for 3D point clouds can be divided mainly into two categories [7]. The first one is conversion-based, which converts the 3D data to some intermediate representations, for example,

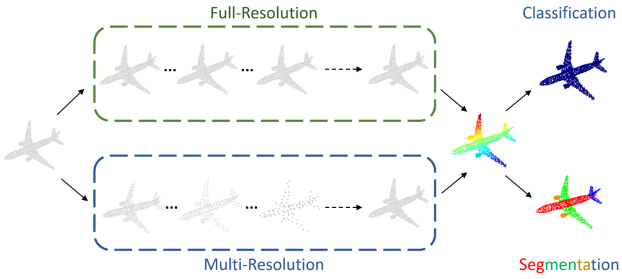

Figure 1. A birdeyes view of our Dense-Resolution Network.

MVCNN [34] projects 3D shapes into multi-view 2D images, and VoxNet [23] transfers point clouds as volumetric grids. The other one is point-based such as PointNet [28], which directly processes points. The point-based approach has become popular due to the introduction of the multilayer perceptrons (MLPs) operation in [28]. The subsequent algorithms [39, 35] adopted MLPs to learn the local features of point clouds using graph context and kernel points.

In order to recognize fine-grained patterns for complex objects or scenes, it is necessary to capture the local spatial context of point clouds. To represent local areas for point clouds, Qi *et al*. [29] and Liu *et al*. [19] apply the Ball Query algorithm [27] to group local points, while Wang *et al*. [39] uses k-nearest neighbors (*knn*) to build point neighborhoods. However, when using these methods, the performance is strongly affected by the areas of their *pre-defined* neighborhoods, *i.e*. the searching radius of a Ball Query, or the k of *knn*. If the area is too small, it cannot cover sufficient local patterns; if too large, the overlap may involve redundancies. DPC [6] proposes an idea of *dilated point convolution* to increase the size of the receptive field without additional computational cost. Unlike previous works, we attempt to adaptively define such a local area for each point *w.r.t*. the density distribution around it, by which the point neighborhood would be more reasonable though requiring less manual intervention and parameter tuning.

Unlike 2D images whose pixels are well-organized in local neighborhoods, learning the feature representations of scattered, unordered, and irregular 3D point clouds are al-

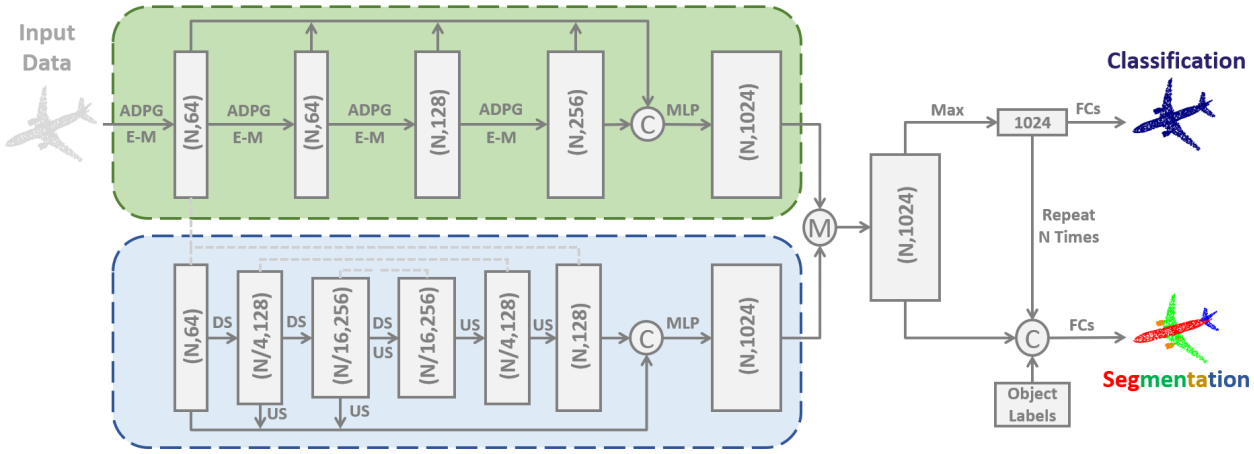

Figure 2. Dense-resolution network architecture. For the FR branch (in green), we learn the full-resolution point cloud features through a series of Error-minimizing modules (denoted as *E-M*, see Section 3.2) involving the Adaptive Dilated Point Grouping method (denoted as *ADPG*, see Section 3.1). For the MR branch (in blue), point features of different resolutions are investigated in a down/up-sampling manner with skip connections (dotted lines). *DS* and *US* represent our down-sampling and up-sampling processes (more details are in Section 4.1 and the supplementary material), respectively. By merging the feature maps (denoted as *M*, see Eq. 4) of the two branches, we manage point cloud classification and segmentation tasks using fully-connected (*FC*) layers. *C* stands for concatenating along channels.

ways challenging. Although one can construct local areas based on the spatial distances between points, the process may accumulate biases from different scales of embedding space and further affect the performance. In addition to feature encoding, an effective mechanism is also required to guide the procedure to learn local features.

Previously, the idea of error feedback has been applied in 2D human pose estimation [4] and image Super-Resolution (SR) [8, 20], in order to regulate the network by compensating the estimated error. To leverage the properties of both error-feedback and CNN training mechanism, unlike the complex error-correcting structures in [15, 30], we propose an error-minimizing module with lower complexity but better performance. Meanwhile, we present a new network architecture, named Dense-Resolution Network (DRNet), for basic 3D point cloud classification and segmentation tasks. By merging feature maps of a Full-Resolution (FR) branch that investigates the full size of the point cloud and a Multi-Resolution (MR) branch that explores different resolutions of the point cloud in a novel fusion method, we can obtain more information for a comprehensive analysis. The main contributions are:

- We propose a novel point grouping method to find neighbors for each point adaptively, considering the density distribution of the neighbors.
- We design an error-minimizing module leveraging the idea of error feedback mechanism to learn the local features of point clouds.
- We introduce a new network to comprehensively represent point clouds from different resolutions.
- We conduct thorough experiments to validate the properties and abilities of our proposals. Our results demonstrate that our approach outperforms state-of-the-art methods on three point cloud benchmarks.

2. Related Work

Local points grouping. Contrary to the pioneer PointNet [28] that relied on global features, subsequent work captured more local features in detail. PointNet++ [29] firstly applied Ball Query, an algorithm for collecting possible neighbors of a particular point through a ball-like searching space centering at a point, to group local neighbors. Similarly, local features learning methods such as [39, 6, 30] use another simple algorithm *knn* gathering nearest neighbors based on a distance metric.

Although Ball Query and *knn* grouping are intuitive, sometimes the size of the neighborhood (*i.e.* the receptive field of the point) is limited due to the range of searching (*i.e.* the radius of query ball, or the value of k). Meanwhile, merely increasing the searching range may involve substantial computational cost. To solve this problem, DPC [6] extended regular *knn* to *dilated-knn*, which gathers local points over a dilated neighborhood obtained by computing the $k \cdot d$ nearest neighbors (d is the dilation factor [46]) and preserving only every d-th nearest point. Moreover, recent works [29, 19, 44] group neighbors through query balls in different scales (*e.g.*, multi-scale grouping) to capture information from various sizes of the local area.

However, the existing methods have some issues in common. On the one hand, the performance of grouping methods highly relies on pre-defined settings. For example, DGCNN [39] provided the results under different k conditions, DPC [6] compared the effects of d values, and PointNet++ [29] discussed the influence of the query ball radius. On the other hand, the grouping methods act on all points without considering each point or object's distinct condition. As far as we are concerned, it is necessary to find an intelligent point-level adaptive grouping method.

Error feedback structure. Previously in 2D computer vision, Carreira *et al*. [4] proposed a framework called Iterative Error Feedback (IEF), which minimized the error loss between current and desired outputs in the back-propagation procedure. In contrast to [4], the methods in [8, 20] complimented the output with a back-projection unit in the forward procedure. For 3D point clouds, PU-GAN [15] leveraged a similar idea for point cloud generation, while [30] presented a structure with specially designed paths for prominent features learning.

Basically, current IEF structures for point clouds are redundant and implicit. Considering the complexity of 3D data, a concise and explicit IEF module is needed. More importantly, an IEF module is expected to serve two purposes in the network: first, to make the actual output approach the desired point clouds representations; second, to help the grouping process form the adaptive point neighborhoods.

Network architecture for point cloud learning. To realize different computer vision tasks using deep learning, many network architectures have been introduced: VGG [33], ResNet [9], *etc*. Besides, some works tried different image resolutions for more clues; for example, the fully convolutional network [21] keeps the full size of an image, deconvolution network [26] steps into lower resolutions, and HRNet [38] shares the features among different resolutions.

As for 3D point clouds, two popular architectures are 1) PointNet++[29], which downsamples the point clouds using Farthest Point Sampling (FPS) and upsamples using Feature Propagation (FP), and 2) a fully convolutional network, which learns point-wise features from multiple embedding space scales, for example, DGCNN [39] dynamically updates the crafted point graph around each point. Different from the above mentioned methods, our approach exploits more clues through dense connections between various resolutions of the point clouds. Furthermore, we investigate the characteristics of multi-resolutional features, and then develop a better merging behavior for the feature maps. In general, our DRNet adaptively encodes the local context from more resolutions of point clouds, by which fine-grained output representations benefit point cloud classification and segmentation tasks.

Algorithm 1: The *forward pass* pipeline of Adaptive Dilated Point Grouping

input: feature map $\mathcal{P}_{N\times c} = [{p_1}^T,\ {p_2}^T,\ \dots,\ {p_N}^T]$ in c-dimensional space.
parameters: the number of neighbors k, and an empirical maximum dilation factor d_{max}.
output: the matrix $\mathcal{I}_{N\times k}$, indices of the selected k neighbors for the point cloud.
for *each point cloud* **do**
  search for the $(k \cdot d_{max})$ candidate neighbors based on $\mathcal{P}_{N\times c}$, get the candidate metric values $E_{N\times(k\cdot d_{max})}$ and the indices $\mathcal{I}_{N\times(k\cdot d_{max})}$;
  learn the dilation factors $\mathcal{D}_N$ based on the metrics $E_{N\times(k\cdot d_{max})}$, where: $d_i \in \mathbb{Z}$, $d_i \in [1, d_{max}]$, $\mathcal{D}_N = [d_1,\ \dots,\ d_i,\ \dots,\ d_N]^T$;
  group the indices $\mathcal{I}_{N\times k}$ of the k neighbors from $\mathcal{I}_{N\times(k\cdot d_{max})}$ based on $\mathcal{D}_N$;
end for

3. Approach

CNN-based learning of 3D data has become more intuitive due to the introduction of multi-layer perceptrons (MLPs) [28] that directly process point clouds. Primarily, an MLP, $\mathcal{M}(\cdot)$, is described as a composite operation of 1-by-1 convolution with a possible batch normalization [11] (BN) and an activation (*e.g*., ReLU) on the feature map.

In addition, recent works [39, 6, 44] craft regional patterns to record more local details via a graph around each point $p_i \in \mathbb{R}^c$, based on both the absolute position of the centroid and relative positions of the neighbors in c-dimensional feature space. Specifically, the crafted graph ($\mathcal{G}$) of the centroid p_i is formulated as: $\mathcal{G}(p_i) = (p_i, p_j - p_i)$; where $\forall p_j \in Ni(p_i)$. Usually, the quality of the information provided by $\mathcal{G}(p_i)$ highly depends on the neighbors, $\forall p_j \in Ni(p_i)$, that are found by the grouping method. Hence, we expect a better grouping method for $\mathcal{G}(p_i)$.

3.1. Adaptive Dilated Point Grouping

The two popular grouping methods *i.e*. Ball Query and k-nearest neighbors (*knn*) (see Section 1) have shortcomings (as analyzed in Section 2), and to overcome these issues, here we propose a novel grouping method named Adaptive Dilated Point Grouping (ADPG), which is shown in Algorithm 1. ADPG aims to generate the indices of neighbors $\mathcal{I}_{N\times k}$ for the points, given a feature map $\mathcal{P}_{N\times c}$ of the point cloud and consists of the following three main procedures.
Searching. The first step of ADPG is searching candidate neighbors for the points. In this paper, we introduce a solution capable of addressing common scales of point cloud data. We define the pairwise Euclidean distances $E_{N\times N}$ in feature space as our metric, which indicates the point

density distribution to a certain extent. As for a $N \times c$ size feature map $\mathcal{P}$, the pairwise Euclidean distances are: $E_{N\times N} = diag(\mathcal{P}\mathcal{P}^T) \cdot \vec{1} + \vec{1}^T \cdot diag(\mathcal{P}\mathcal{P}^T)^T - 2\mathcal{P}\mathcal{P}^T$, where $\vec{1}$ means a $1 \times N$ row vector of ones, and $diag(\cdot)$ forms a $N \times 1$ column vector whose entries are the N diagonal elements of a $N \times N$ square matrix.

According to the calculated distances metric, we can easily identify the $k \cdot d_{max}$ candidate nearest neighbors of each point. In our implementation, we sort the rows of $E_{N\times N}$ in ascending order, and retain the metric values and indices of the first $k \cdot d_{max}$ elements. Therefore, the elements with the smallest $k \cdot d_{max}$ values in each row of $E_{N\times N}$ are identified as candidate neighbors for each point. Meanwhile, the metric values and indices of the searched candidate neighbors are recorded as $E_{N\times(k\cdot d_{max})}$ and $\mathcal{I}_{N\times(k\cdot d_{max})}$, respectively. Besides, our implementation is also flexible; that is, the choices for metrics (*e.g.*, density or geometric similarities) and searching techniques (*e.g.*, FLANN [25] for the sake of efficiency in large-scale point cloud data) can be easily integrated as needed.

Learning. In order to construct a dilated neighborhood for each point adaptively, it is necessary to determine a dilation factor [46] for each point based on known information of its candidate neighbors. In practice, we learn the dilation factors based on $E_{N\times(k\cdot d_{max})}$ and CNN-related operations.

To be specific, we apply an MLP ($\mathcal{M}$) and a sigmoid function (σ) to the metric values of candidates $E_{N\times(k\cdot d_{max})}$, in order to summarize the information of the point distribution of local areas. Then, a projection function $\mathcal{J}$ (*e.g.*, *linear function*) can map the values to the expected numerical scale. Finally, we take a scale function $\mathcal{S}$ (*e.g.*, *round* to assign a dilation factor, $\mathcal{D}_N$[1], for each point according to the summarized information:

$$\mathcal{D}_N = \mathcal{S}\Big(\mathcal{J}\Big(\sigma\big(\mathcal{M}(E_{N\times(k\cdot d_{max})})\big)\Big)\Big). \tag{1}$$

Grouping. As each point has a corresponding dilation factor, we pick up every d_i-th index of candidate indices $\mathcal{I}_{N\times(k\cdot d_{max})}$ to form the selected k neighbors for each point. Following a behavior similar to [6], we obtain the final indices of local point groups $\mathcal{I}_{N\times k}$.

3.2. Error-minimizing Module

Following the ADPG method, each point gathers a group of neighbors with a larger receptive field. As stated, we apply the crafted graph $\mathcal{G}$, *i.e.* the absolute position of a centroid and relative positions of the neighbors, to encode the high-dimensional features over each neighborhood. Further projected by an MLP (with c' filters), the information of a local graph centering at p_i, is represented as:

$$f_{\mathcal{G}_i} = \mathcal{M}\big(\mathcal{G}(p_i)\big) = \mathcal{M}\big((p_i, p_j - p_i)\big), \tag{2}$$

where $\forall p_j \in ADPG(p_i)$ and $f_{\mathcal{G}_i} \in \mathbb{R}^{c'\times k}$.

Usually, a max-pooling function is applied over the k neighbors of each crafted local graph to aggregate the local context as the centroid's feature representation. However, possible bias exists in process: on the one hand, the local graphs lack geometric regularization from the initial 3D space; and on the other hand, the max-pooled features only retain prominent outlines while discarding local details in embedding space. In this case, the Iterative Error Feedback (IEF) mechanisms idea helps avoid bias accumulation during the high-dimensional feature learning process.

Let us assume that the local graph $f_{\mathcal{G}_i}$ indeed embeds the full information about the neighborhood, it would be possible to restore the input p_i through a back-projection process $\mathcal{B}(\cdot)$. Practically, we realize the $\mathcal{B}(\cdot)$ operation through a shared 1-by-k convolution followed by BN and ReLU, over the local graphs. Intuitively, this operation acts to aggregate the nodes based on learned weights of the edges in the graph, which implicitly simulates a reverse process of crafting the graph. Therefore, the back-projected feature $f_{\mathcal{B}_i}$ is formulated as: $f_{\mathcal{B}_i} = \mathcal{B}(f_{\mathcal{G}_i})$; where $f_{\mathcal{B}_i} \in \mathbb{R}^c$.

Consequently, an error feature $f_{\mathcal{E}_i}$ is defined as the difference between the original input p_i and back-projected feature $f_{\mathcal{B}_i}$. In contrast to the methods in [15, 30, 8, 20] that correct the error by extra computations in the forward pass, we use additional ℓ_2 loss to minimize the error, $f_{\mathcal{E}_i} = f_{\mathcal{B}_i} - p_i$, during the back-propagation pass:

$$\mathcal{L}_{er} = ||f_{\mathcal{E}_i}||_2. \tag{3}$$

The loss in Equation 3 can constrain the feature learning during training by forcing the back-projected feature $f_{\mathcal{E}_i}$ to approach the original input p_i inside of the module. Following such a regularization, the error and bias in the output representations can be alleviated, especially during the early stages of training. Meanwhile, compared with the regular cross-entropy loss for the whole network, each error-minimizing module's loss can provide more clues for the ADPG in corresponding feature space.

3.3. Dense-Resolution Network Architecture

Although the ADPG method and the error-minimizing module seem promising for local feature learning of point clouds, we still need a robust network architecture to leverage the potential offered by both. The architecture of our network is presented in Figure 2.

Full-resolution branch. We adopt the idea of basic fully convolutional architecture as the full-resolution (FR) branch of our network. The benefits can be retained based on two aspects; 1) there remains a consistent number of points in different scales of embedding space during feature learning progress; 2) it retains the per-point feature without any confusion caused by the numerical approximation in upsam-

[1] More implementing details are in the supplementary material.

pling. Therefore, we expect this structure to learn comprehensive representations for point-wise features.

Specifically, the FR branch consists of the proposed error-minimizing modules in a cascaded form, which progressively learn the feature representation of each point from its adaptive neighborhood formed by ADPG in different scales of embedding space. In order to acquire a global knowledge about the abstract embedding space, the learned features from different scales are concatenated and aligned to form the output $\mathcal{F}_{FR}$ of the FR branch.

Multi-resolution branch. Meanwhile, there is a limitation of $\mathcal{F}_{FR}$: it lacks channel-wise clues about semantic/shape-related information since the FR branch mainly focuses on point-wise context. To overcome this issue, we capture additional features from more resolutions of point clouds. Therefore, we propose the multi-resolution (MR) branch, a light-weight down/up-sampling structure, to investigate the lower resolutions of point clouds. Contrary to competing methods, the propagated features and skip links are densely connected to enhance the relations between multiple point cloud resolutions and feature embedding scales. The output $\mathcal{F}_{MR}$ of the MR branch captures thorough channel-wise information about the point clouds.

Features merging. To leverage the information gathered from both FR and MR branches, it is necessary to find a reasonable merging technique for the two feature maps, *i.e*. $\mathcal{F}_{FR}$ and $\mathcal{F}_{MR}$. Usually, CNNs combine the feature maps by concatenation, summation, or multiplication. These regular operations treat the feature maps equally, without taking their properties into account. Instead, we prefer merging the FR and MR outputs in a unique manner.

Given the advantages of FR and MR branches that we analyzed before, $\mathcal{F}_{FR}$ is applied as the basis of per-point feature representation. In addition, the channel-wise information of $\mathcal{F}_{MR}$ is derived to enhance $\mathcal{F}_{FR}$. Empirically, we use a max-pooling and an MLP to summarize the knowledge of $\mathcal{F}_{MR}$ channels. After a sigmoid activation σ, the channel-wise enhancement on the per-point context of $\mathcal{F}_{FR}$ can be realized by multiplication. The final output of our dense-resolution (DR) network follows:

$$\mathcal{F}_{DR} \;=\; \mathcal{F}_{FR} \;\times\; \sigma\Big(\mathcal{M}\big(\max_{N}(\mathcal{F}_{MR})\big)\Big). \tag{4}$$

Loss function. Based on the output feature map ($\mathcal{F}_{DR}$), the fully-connected (FC) layers regress the confidence scores for all possible categories. In addition to the basic cross-entropy loss ($\mathcal{L}_{ce}$), the weighted losses of the error-minimizing modules are incorporated. For the DRNet with M error-minimizing modules in its FR branch, by applying Equation 3 and the hyper-parameter w_i as weight, the overall loss is formulated as:

$$\mathcal{L} = \mathcal{L}_{ce} + \sum_{i=1}^{M} w_i \cdot \mathcal{L}_{er_i}. \tag{5}$$

4. Experiments

In this section, our implementation details are provided, including network parameters, training settings, datasets, *etc*. By comparing the experimental results with other state-of-the-art methods, we analyze performance quantitatively. Further, we present ablation studies and visualizations to illustrate the properties of our approach.

4.1. Implementation

Network details. The FR branch of our DRNet is a series of error-minimizing modules extracting features at different scales of embedding space: *i.e*. 64, 128, and 256, as in Figure 2. Empirically, we adopt $k = 20$ and $d_{max} = 5$ as in [39, 6]. The FR output $\mathcal{F}_{FR}$ is an MLP projected concatenation of the modules' outputs. For the MR branch, we apply the widely-used farthest point sampling (FPS) and feature propagation (FP) [29, 19, 18] for downsampling and upsampling, respectively. Further, single-layer MLPs are used for channel alignment together with the mentioned operations. The MR branch starts from the first output of FR in N size; after that, two lower resolutions: N/4 and N/16, are investigated through the regular *knn* and local graph encoding as Equation 2. Different from other approaches, more propagated features and dense skip connections are employed to enhance the relations between different point resolutions and feature spaces. Compared with the FR, the MR branch[2] is light-weight due to the fewer scales of embedding space, the limited number of points, and the operations with fewer learnable weights.

The output $\mathcal{F}_{DR}$ is obtained by following Equation 4. For the classification task, we apply a max-pooling function and Fully Connected (FC) layers to regress confidence scores for all possible categories. In terms of the segmentation task, we attach the max-pooled feature to each point feature of $\mathcal{F}_{DR}$ and further predict each point's semantic label with FC layers being applied.

For the loss function, empirically, a larger weight is set for the first error-minimizing module, *i.e*. w_1, since its output affects both branches and constrains the network learning initially. In contrast, the weights for other modules can be smaller since they are less critical. Although the additional loss is involved, cross-entropy loss still contributes the most to the training[2]. We implement the project with PyTorch and Python; all experiments are conducted on Linux and GeForce RTX 2080Ti GPUs.[3]

Training strategy. For classification, Stochastic Gradient Descent (SGD) [22] with a momentum of 0.9 is adopted as the optimizer. The learning rate decreases from 0.1 to 0.001 by cosine annealing [22] during the 300 epochs. For

[2]More information about the implementation is provided in the supplementary material.

[3]The code and models are available at `https://github.com/ShiQiu0419/DRNet`

	overall mIoU	air plane	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	moto bike	mug	pistol	rocket	skate board	table
# shapes	16881	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
PointNet [28]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
A-SCN [42]	84.6	83.8	80.8	83.5	79.3	90.5	69.8	91.7	86.5	82.9	**96.0**	69.2	93.8	82.5	62.9	74.4	80.8
SO-Net [14]	84.6	81.9	83.5	84.8	78.1	90.8	72.2	90.1	83.6	82.3	95.2	69.3	94.2	80.0	51.6	72.1	82.6
PointNet++ [29]	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
PCNN [1]	85.1	82.4	80.1	85.5	79.5	90.8	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
DGCNN [39]	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	**63.5**	74.5	82.6
P2Sequence [17]	85.2	82.6	81.8	87.5	77.3	90.8	77.1	91.1	86.9	83.9	95.7	70.8	94.6	79.3	58.1	75.2	82.8
SpiderCNN [43]	85.3	83.5	81.0	87.2	77.5	90.7	76.8	91.1	87.3	83.3	95.8	70.2	93.5	82.7	59.7	75.8	82.8
PointASNL [44]	86.1	84.1	84.7	87.9	**79.7**	**92.2**	73.7	91.0	87.2	84.2	95.8	**74.4**	**95.2**	81.0	63.0	76.3	83.2
RS-CNN [19]	86.2	83.5	84.8	**88.8**	79.6	91.2	**81.1**	91.6	88.4	**86.0**	**96.0**	73.7	94.1	**83.4**	60.5	**77.7**	83.6
Ours	**86.4**	**84.3**	**85.0**	88.3	79.5	91.2	79.3	**91.8**	**89.0**	85.2	95.7	72.2	94.2	82.0	60.6	76.8	**84.2**

Table 1. Part segmentation results (mIoU(%)) on the *ShapeNet Part* dataset.

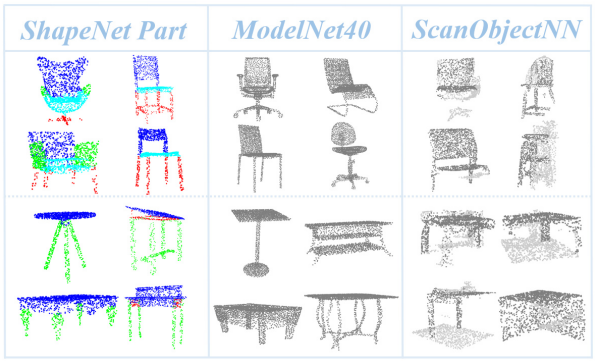

Figure 3. Examples from the experimental datasets. The upper row shows the point clouds labeled as *Chair* from the three datasets, while the lower row presents *Table*. Particularly, *ScanObjectNN* dataset contains background points, which are in a lighter color.

method	input type	#points	accuracy
PointNet [28]	coords	$1k$	89.2
A-SCN [42]	coords	$1k$	90.0
PointNet++ [29]	coords	$1k$	90.7
SO-Net [14]	coords	$2k$	90.9
PointCNN [16]	coords	$1k$	92.2
PCNN [1]	coords	$1k$	92.3
SpiderCNN [43]	coords+norms	$1k$	92.4
PointConv [40]	coords+norms	$1k$	92.4
P2Sequence [17]	coords	$1k$	92.6
DensePoint [18]	coords	$1k$	92.8
RS-CNN [19]	coords	$1k$	92.9
DGCNN [39]	coords	$1k$	92.9
KP-Conv [35]	coords	$7k$	92.9
PointASNL [44]	coords	$1k$	92.9
Ours	coords	$1k$	**93.1**

Table 2. Overall classification accuracy (%) on *ModelNet40* dataset. ($coords$: 3D coordinates, $norms$: surface normal vectors of the points, k:$\times 2^{10}$)

segmentation, we exploit Adam [13] optimization for 200 epochs of training. The learning rate begins at 0.001 and gradually decays with a rate of 0.5 after every 20 epochs. The batch size for both tasks is 32. Besides, training data is augmented with random scaling and translation; the overall loss follows Equation 5. Part segmentation is evaluated with a ten-votes strategy used by recent approaches [28, 29, 19].

Datasets. We test our approach on two main tasks: point cloud segmentation and classification. The ShapeNet Part dataset [45] is used to predict the semantic class (*part label*) for each point of the object. In addition, the synthetic ModelNet40 [41] dataset and the real-world ScanObjectNN [36] dataset are used to identify the category of the object. Figure 3 presents some examples from the datasets.

- **ShapeNet Part.** The dataset has 16,881 object point clouds in 16 categories, where each point is labeled as one of 50 parts. As the primary dataset for our experiments, we follow the official data split [5]. We input the 3D coordinates of 2048 points for each point cloud and feed the object label before FC layers during training. In terms of the metric for evaluation, we adopt Intersection-over-Union (*i.e.* IoU). The IoU of the shape is calculated by the mean value of IoUs of all parts in that shape, and mIoU (*i.e.* mean IoU) is the average of IoUs for all testing shapes.

- **ModelNet40.** It is a popular dataset because of regular and clean shapes. There are 12,311 meshes in 40 classes, with 9,843 for training and 2,468 for testing. Corresponding point clouds are generated by uniformly sampling from the surfaces, translating to the origin, and scaling within a unit sphere [28]. In our case, only the 3D coordinates of 1024 points for each point cloud have been used.

- **ScanObjectNN.** This real-world object dataset is recently published. Although it has over 15,000 objects in only 15 categories, it is practically more challenging due to the background complexity, object partiality, and different deformation variants. We conduct the experiment using its most challenging variant, *PB_T50_RS*, with background points.

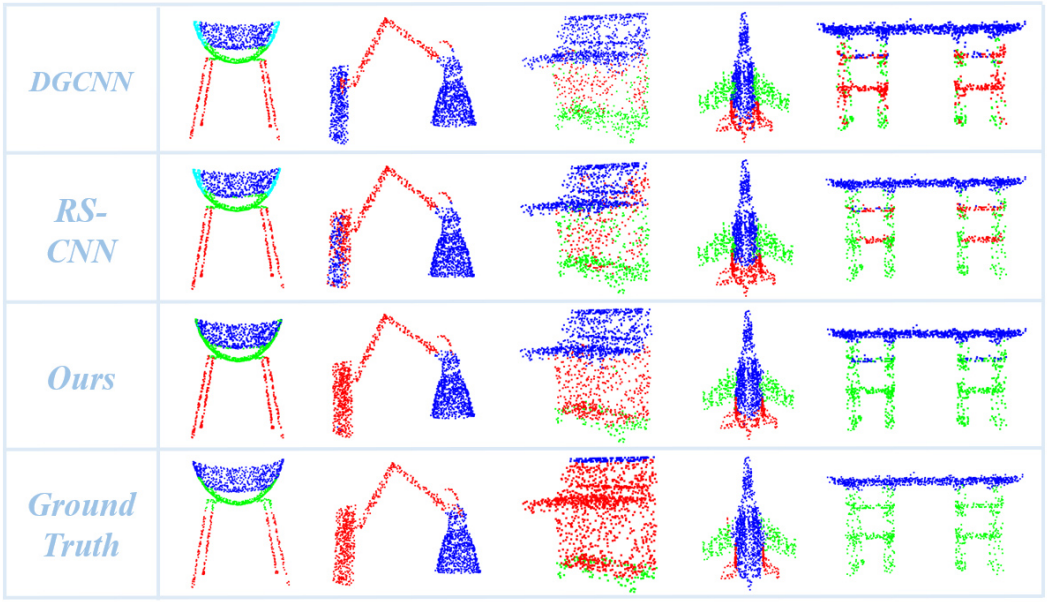

Figure 4. Examples of the part segmentation results. (DGCNN: [39], RS-CNN: [19])

	overall acc.	avg class acc.	bag	bin	box	cabinet	chair	desk	display	door	shelf	table	bed	pillow	sink	sofa	toilet
# shapes	-	-	298	794	406	1344	1585	592	678	892	1084	922	564	405	469	1058	325
3DmFV [2]	63.0	58.1	39.8	62.8	15.0	65.1	84.4	36.0	62.3	85.2	60.6	66.7	51.8	61.9	46.7	72.4	61.2
PointNet [28]	68.2	63.4	36.1	69.8	10.5	62.6	89.0	50.0	73.0	**93.8**	72.6	67.8	61.8	67.6	64.2	76.7	55.3
SpiderCNN [43]	73.7	69.8	43.4	75.9	12.8	74.2	89.0	65.3	74.5	91.4	78.0	65.9	69.1	80.0	65.8	90.5	70.6
PointNet++ [29]	77.9	75.4	49.4	**84.4**	31.6	77.4	91.3	**74.0**	79.4	85.2	72.6	72.6	75.5	**81.0**	**80.8**	90.5	**85.9**
DGCNN [39]	78.1	73.6	49.4	82.4	33.1	**83.9**	91.8	63.3	77.0	89.0	79.3	**77.4**	64.5	77.1	75.0	91.4	69.4
PointCNN [16]	78.5	75.1	57.8	82.9	33.1	83.6	**92.6**	65.3	78.4	84.8	**84.2**	67.4	**80.0**	80.0	72.5	**91.9**	71.8
Ours	**80.3**	**78.0**	**66.3**	81.9	**49.6**	76.3	91.0	65.3	**92.2**	91.4	83.8	71.5	79.1	75.2	75.8	**91.9**	78.8

Table 3. Classification results (%) on *ScanObjectNN* dataset.

4.2. Results

Segmentation. Table 1 shows the results of related works reported in overall mIoU, which is the most critical evaluation metric on the ShapeNet Part dataset. On the whole, our network achieves 86.4% and outperforms other state-of-the-art algorithms based on similar experimental settings. For evaluations inside each class, we surpass others in five out of 16 categories. Especially in categories with a large number of samples, *e.g.*, airplane, chair, or table, we perform even better (two out of these three classes) than others. In Figure 4, we provide some samples of our part segmentation results comparing with DGCNN [39] and RS-CNN [19].

Classification. Table 2 presents the overall accuracy of the classification on the synthetic object dataset: ModelNet40. Specifically, we achieve 93.1% for overall classification accuracy and exceed other state-of-the-art results with similar input. Essentially, our method performs better than others using more input points or features.

Table 3 presents our results on the ScanObjectNN dataset, which contains practical scans of real-world objects as Figure 3 indicates. To be concrete, both overall accuracy 80.3% and average class accuracy 78.0% of our approach are significantly higher than all results on its official leaderboard [10]. Typically, we lead in four (*bag, box, display, and sofa*) out of the 15 categories. The inference time of our basic classification model running on a single GeForce RTX 2080Ti GPU is about *19.2ms*.

4.3. Ablation Studies

Visualization of learned dilation factors. Figure 5 illustrates the effects of our ADPG method, where the color of the points corresponds to the learned dilation factor. Intuitively, the advantages of ADPG can be observed from two aspects: Firstly, for each point cloud, ADPG tends to assign larger dilation factors to points that have relatively sparse local point distributions (*e.g.*, on corners/boundaries/edges) because they need larger neighborhoods for more comprehensive local feature learning. Secondly, within the cascaded structure, ADPG regulates the points' dilation factors in deep layers and turns out to have smaller dilation factors in dense local distribution (*e.g.*, on flat surfaces/central areas), most probably to constrain the neighborhoods and

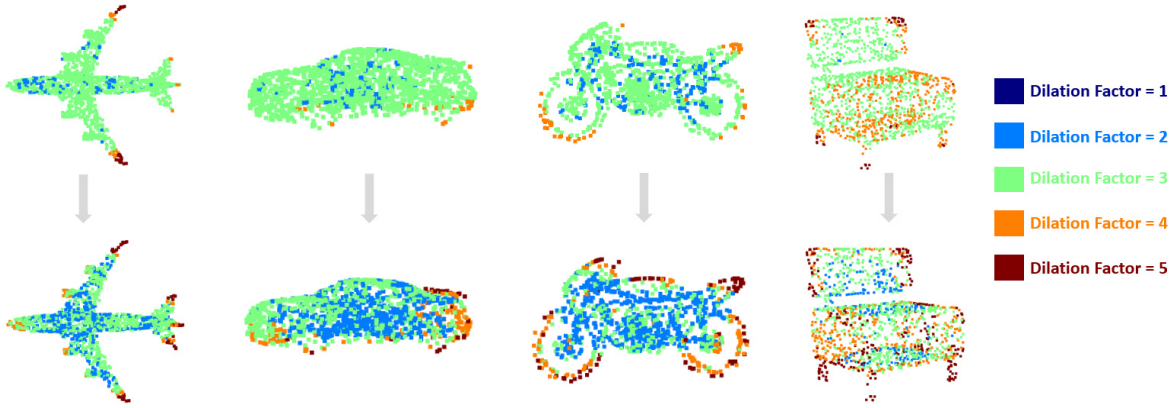

Figure 5. Learned dilation factors by the ADPG method. For each point cloud, ADPG assigns larger dilation factors for the points in sparse areas. As the network goes deeper, ADPG regulates the dilation factors of the points. (First-row: the learned dilation factors in a shallow layer of our network. Second-row: in a deep layer.)

model	Network	ADPG	E-M module	overall mIoU
0	FR	-	-	85.2
1	FR	-	✓	85.4
2	FR	✓	✓	85.6
3	MR	-	-	84.9
4	MR	✓	✓	85.3
5	***DR***	✓	✓	**86.0**

Table 4. Ablation study about the effects of different network components on ShapeNet Part (%). (FR: Full-Resolution branch only, MR: Multi-Resolution branch only, DR: Dense-Resolution Network, ADPG: Adaptive Dilated Point Grouping method, E-M module: Error-minimizing module for local points.)

model	Network	$\mathcal{F}_{mer}$	overall mIoU
0	FR	$\mathcal{F}_{FR}$	85.6
1	MR	$\mathcal{F}_{MR}$	85.3
2	DR	$Concat(\mathcal{F}_{FR}, \mathcal{F}_{MR})$	85.7
3	DR	$\mathcal{F}_{FR} + \mathcal{F}_{MR}$	85.6
4	DR	$\mathcal{F}_{FR} \odot \mathcal{F}_{MR}$	85.6
5	***DR***	$\mathcal{F}_{DR}$	**86.0**

Table 5. Ablation study about the different forms of merged feature $\mathcal{F}_{mer}$ on ShapeNet Part (%). (FR: Full-Resolution branch only, MR: Multi-Resolution branch only, DR: Dense-Resolution Network, $\mathcal{F}_{FR}$: the output of FR, $\mathcal{F}_{MR}$: the output of MR, $\odot$: element-wise multiplication, $\mathcal{F}_{DR}$: merging as in Equation 4.)

avoid outliers. Unlike regular *knn*/Ball Query/*dilated-knn*, which defines a limited and fixed neighborhood for all points in all layers, our ADPG works adaptively and reasonably as expected.

Effects of components. We conduct an ablation study about the effects of the network components: the architecture, grouping method, and the error-minimizing module. We run tests on the ShapeNet Part dataset under the same settings, and Table 4 presents the results. Comparing model 1&2 to model 0 and model 4 to model 3, we observe that the error-minimizing module with ADPG applied can significantly improve the part segmentation's network performance. Although the multi-resolution branch is not able to learn the features as comprehensively as a full-resolution branch does, we can take advantage of both by combining them as a dense-resolution network (model 5).

Merging the feature maps. Both FR and MR have properties as mentioned, so we need to find an effective way to unify the benefits. We test simple ways of merging the features of $\mathcal{F}_{FR}$ and $\mathcal{F}_{MR}$, *i.e.* concatenating them in channel-wise, adding and multiplying them in element-wise. Comparing the results of model 2&3&4 to model 0 in Table 5, we observe that the simple ways of merging may not improve performance. In contrast, channel-wise enhancing of the $\mathcal{F}_{FR}$ using $\mathcal{F}_{MR}$ (model 5) can improve a bit because of the reasons explained in Section 3.3. With ten-votes testing, the overall mIoU can boost to 86.4%.

5. Conclusion

In this work, we propose a Dense-Resolution Network for point cloud analysis, which leverages information from different resolutions of the point clouds. Specifically, the Adaptive Dilated Point Grouping method is introduced to realize a flexible point grouping based on the density distribution. Moreover, an error-minimizing module and corresponding loss are presented to capture local information and guide the training network. We conduct experiments and provide ablation studies on both point cloud segmentation and classification benchmarks. The experimental results outperform competing state-of-the-art methods on ShapeNet Part, ModelNet40, and ScanObjectNN datasets. The quantitative reports and qualitative visualizations demonstrate the advantages of our approach.

References

[1] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.

[2] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152, 2018.

[3] François Blais et al. Review of 20 years of range sensor development. *Journal of electronic imaging*, 13(1):231–243, 2004.

[4] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4733–4742, 2016.

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[6] Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. Dilated point convolutions: On the receptive field of point convolutions. *arXiv preprint arXiv:1907.12046*, 2019.

[7] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[8] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1664–1673, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] HKUST-VGD. 3d scene understanding benchmark. `https://hkust-vgd.github.io/benchmark/`, 2020. Accessed: 2020-08-20.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[12] Michel Jaboyedoff, Thierry Oppikofer, Antonio Abellán, Marc-Henri Derron, Alex Loye, Richard Metzger, and Andrea Pedrazzini. Use of lidar in landslide investigations: a review. *Natural hazards*, 61(1):5–28, 2012.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018.

[15] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: A point cloud upsampling adversarial network. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[16] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018.

[17] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8778–8785, 2019.

[18] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[19] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[20] Zhi-Song Liu, Li-Wen Wang, Chu-Tak Li, and Wan-Chi Siu. Hierarchical back projection network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[23] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[24] Niloy J Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas. Registration of point cloud data from a geometric optimization perspective. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 22–31. ACM, 2004.

[25] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[26] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[27] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on

point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[30] Shi Qiu, Saeed Anwar, and Nick Barnes. Geometric back-projection network for point cloud classification. *arXiv preprint arXiv:1911.12885*, 2019.

[31] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.

[32] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[35] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[36] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[37] George Vosselman, Sander Dijkman, et al. 3d building model reconstruction from point clouds and ground plans. *International archives of photogrammetry remote sensing and spatial information sciences*, 34(3/W4):37–44, 2001.

[38] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *arXiv preprint arXiv:1908.07919*, 2019.

[39] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019.

[40] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

[41] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[42] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018.

[43] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[44] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5589–5598, 2020.

[45] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

[46] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.